HW1: Mid-term assignment

I. Oliveira, v2023-03-21

Objectives

Develop a **multi-layer web application**, in Spring Boot, supplied with automated tests.
This is an **individual** assignment.

Required elements

Project scope

Your application should provide details on **air quality** for a certain region/city, for the current day and forecasts for upcoming days[1]. The air quality can be characterized with different metrics, such as particles in suspension (PM10,...) or gases (CO, Ozone,...).
Variations on the theme are acceptable and welcome, if they are related to public safety surveillance (e.g.: pollen concentration, meteorology hazard alerts, etc.)

The project must include **different components** (Figure 1)**:**

1. A minimalist **web app/page** which allows users to enter/select a **location** and query its air quality values (it can be more than one page but keep it simple).
2. **Integration with external source(s)**. Actual air quality data should be fetched from a remote service. That is, your backend services will act as a client for a third-party API[2], which you will access to obtain the values.
3. The solution should also implement internally a (in-memory) *cache*[3]*,* with the latest results (from the external source) copied locally; if the same data is requested (i.e., same city and day/s), you should use the *cached* values. The *cache* should define a *time-to-live* policy and produce some basic statistics about its operation (count of requests, *hits/misses),* which you do not need to save persistently [you may use a data structure, like a HashMap, or in-memory database].
4. Your own **API (REST)** that can be invoked by external clients. Your API should allow one to programmatically interrogate (your) backend for useful air quality data (e.g.: by location, by days,...). Your API should also provide basic statistics on the use of the (internal) *cache*.
5. [Recommended] Use some logger support so that your solution produces a useful log (of actions/events).

[1] Data availability depends on the data sources you will use.
[2] There are several API available in the Internet to get air quality data. Make your own survey and assessment; then choose one that you find easy to use to integrate in your project.
[3] This should be a simple cache data structure, implemented by you; you should not use the Spring framework services or other libraries for caching (e.g.: JCache, Spring Boot Cache Manager).

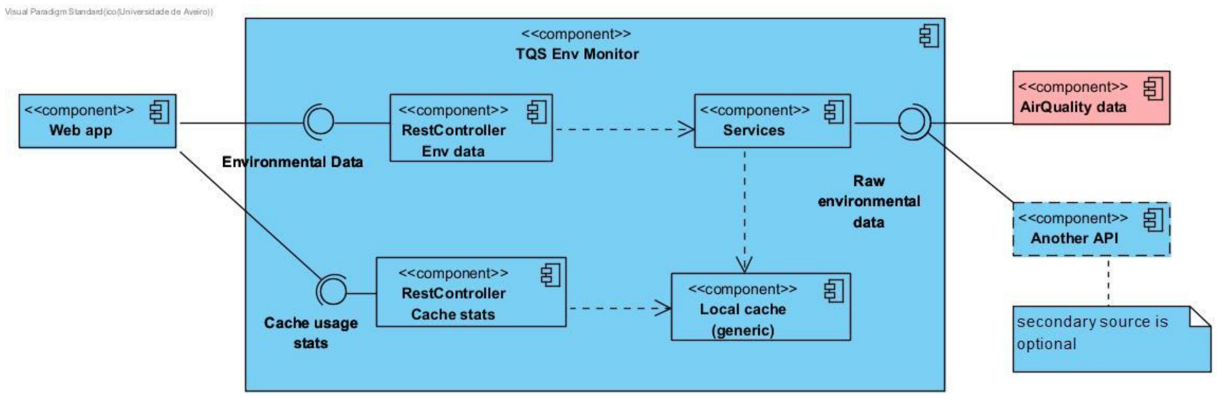

Figure 1: Expected componentes.

Technologies stack

The solution should be based on Spring Boot for the services/backend (The "TQS Env Monitor" component in Figure 1). The web (presentation) layer can be implemented with any HTML/JavaScript framework or using a templating system that integrates with Spring Boot (e.g.: thymeleaf).

Tests to implement

The project should include different types of tests:

A) Unit tests (suggestion: tests on *cache* behavior; tests on "validators", "converters" or "utils" in general, if applicable).
B) Service level tests, with dependency isolation using *mocks* (suggestion: test with isolation from external data provider).
C) Integration tests on your own API (suggestion Spring Boot MockMvc and/or REST-Assured).
D) Functional testing (on the web interface). Suggested technology: Selenium WebDriver.

Quality metrics

The project should include code quality metrics (e.g., from Sonar Qube). To do this, you should also implement:

E) Integration of analysis with Sonar Qube (or Codacy). Note: If the Git project is public, it can be analyzed in SonarCloud.

Extra points (optional)

For extra points:

F) The project may use more than one external source (remote API) and switch between one and the other, either by configuration, or upon the (un)availability of the sources.
G) Continuous Integration pipeline with the automation of testing and static code analysis (e.g.: GitHub Actions, GitLab CI/CD).

Submission

The submission consists of a brief report and a Maven-based code project.

1. A brief **Technical Report** should **explain the strategy** that was adopted (your options as a developer) and offer **evidence** of the results obtained (e.g.: which testes per testing level, screenshots of the representative steps, (small) code snippets of the key parts, screenshots with the test results, etc.). The results of the Sonar Qube dashboard should be included (and discussed). [A report template is available.]
2. The **code project**, committed to your TQS personal Git repository (folder /HW1).

   Besides the code, be sure to include in the repository a short video with a demonstration of your solution (or a link to the video, instead, if the video is a large file).
3. **Oral presentation**. To be scheduled.